Cloud Service in Stock Trading Game: Service Virtualization, Integration and Financial Application

Chin-Wei Wang
Institute of Technology Management
National Chung Hsing University
Taichung City, Taiwan
tommy38382003@yahoo.com.tw

Shuchih Ernest Chang*
Institute of Technology Management
National Chung Hsing University
Taichung City, Taiwan
eschang@dragon.nchu.edu.tw

***Abstract*—More and more cloud services emerging because of increased demand, but there is significant room for improvement. We propose architecture to integrate public cloud services for reducing costs. We consider three issues: (1) The general cloud storage services are unsuitable for large and complex data. A NoSQL datastore, the most common type of database for big data, can process operational information. (2) Integrating several cloud services into centralized systems or applications is challenging. (3) Enterprises often use their own equipment, which accounts for a large part of their costs. This study aims to construct cross-service architecture via free cloud-based NoSQL datastores for virtualizing and integrating needed computing resources and services. We build a stock trading game to test the feasibility and performance of the proposed architecture which can reduce equipment costs by virtualizing computing resources through various cloud-based services.**

***Keywords—NoSQL* datastore, cloud service integration, virtualization, service-oriented architecture**

I. INTRODUCTION

Cloud computing and storage technology has become more mature, and business demand for cloud services is increasing. The risk of equipment failure is problematic for businesses. Equipment can be replaced or repaired; however, restoring services requires significant time and manpower. If severe data loss occurs, it would be difficult to rectify the situation.

Many popular cloud services such as IBM, Google Drive, and Dropbox provide data storage and datastore services. If companies simply upload data, the structure and ease of use will be low. Companies always attempt to increase revenue and reduce costs. Equipment acquisition and maintenance are often the largest part of total costs. Indeed, reduction of service costs in a stable and secure environment has to be considered carefully.

In business, virtualization is an unstoppable trend. Virtualization can change response capabilities during crises, reduce costs and manpower requirements in computer facilities, and better protect services and information. Many cloud computing facilities provide 24-h security, UPS systems, and temperature control as well as shock, fire, and antitheft protection. UPS system is a power supply that includes a battery to maintain power in the event of a power outage. In addition, companies that provide cloud storage and control have more technical expertise than most businesses [1]. Therefore, storing data on the cloud is more secure.

The objective of this study is to build a cross-service architecture using free resources to allow services to run directly on the cloud through virtualization. We construct a cloud-based NoSQL database to store and retrieve data systematically. In addition, we integrate several cloud resources and create integrated cloud services. Not only store the information on the cloud but also cloudify all the foreground and background of our system to implement the concept of service virtualization and integration. The proposed architecture allows users to use the data in each cloud resource. The proposed architecture can also be applied to existing sites. From a business perspective, this architecture can reduce manpower and operation costs. It could provide a more secure and comprehensive good strain at the time of any crisis.

II. RESEARCH BACKGROUND

Our goal is to provide cloud architecture using free cloud data storage space resiliently. In our research, we used three different free cloud spaces. The main elements of the system design are described in the following sections:

A. Cloud Service

Cloud services are available to users on demand via the Internet. The purpose of cloud services is to reduce cost, simplify IT management, and improve productivity through automation and standardization [2].

Typically, cloud services are designed to provide easy, scalable access to applications, resources, and services [3]. Our objective is to design a scalable cloud-based service that can be accessed via the Internet.

B. Cross Services

Many cloud services are available. Each service has its own database format. The provision of cross services between different cloud service providers has become an important issue. Here, we attempt to integrate several cloud services into a single system. By using cross services, businesses can avoid "putting all their eggs in one basket." The cross services feature also can confirm that the proposed architecture could apply to many cloud services online.

We use several application programming interfaces (API) to connect and send requests to the cloud services. On the

*Corresponding author.

control side, the system can distribute and determine data storage and transmission through an agent.

C. Service-Oriented Architecture (SOA)

SOA is a system architecture model. The main concept of SOA is the combination of a set of software components for business requirements. SOA technology has become increasingly important, and companies can use an SOA-based system to integrate dispersed functions and recreate services [4].

At present, many SOA resources are freely available. SOA is a good solution for individual developers to utilize services as fundamentals for developing according to its free existing services. SOA allows easy integration of heterogeneous systems. It also increases re-use of code. We use many existing free APIs and open sources to accelerate the development of the services which is used to transmit data to cloud databases.

D. Cloud Service APIs

Cloud service APIs are used to build cloud applications. Most cloud services are accessed via an API. A good API provides all necessary building blocks, which makes it easier to develop a program. At present, many APIs are freely available. In this study, we use MongoDB, Datastore, and HBase REST APIs to connect to MongoDB, Google Cloud Datastore, and Hadoop HBase. Using these APIs, we can communicate with a cloud datastore. They also allow us to store and query data.

E. Big Data

Big data applications, in the no structural data premise, are becoming increasingly common. However, traditional relational databases are difficult to expand, which makes them unsuitable for big data applications. Unlike highly structured relational databases, NoSQL databases are unstructured and provide a framework for high-performance and agile processing. High scalability and flexibility are important characteristics of NoSQL databases [5]. In this study, we use a NoSQL datastore, which allows us to process large data, such as stock market information.

III. Architecture & Design

Most cloud services provide storage space, which is normally used to store files and documents (Fig. 1). However, some cloud services provide database services (e.g., Google Cloud Datastore), which differ from general storage services.

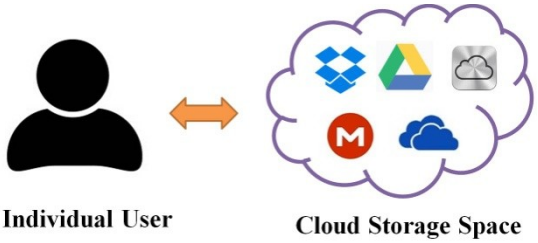

Fig. 1. The architecture of general cloud storage using.

Businesses cannot use general cloud storage to process daily transaction data. Typically, businesses rely on in-house systems developed and maintained by an internal IT department to handle daily operations. IT departments must also control the temperature in computer facilities and prevent equipment failure. Developing and maintaining in-house servers incurs significant operational costs.

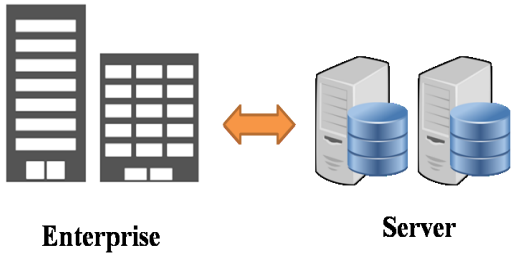

Fig. 2. The architecture of data storage in enterprise operations.

In this study, we focus on free cloud services to build a cross-platform application. The design pattern used to organize and utilize different cloud resources for the proposed architecture is shown in Fig. 3.

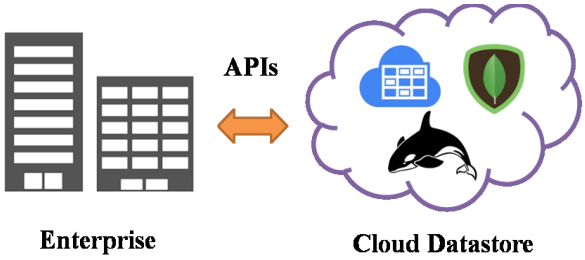

Fig. 3. The architecture of cloud datastore access in an enterprise.

External free APIs are used to replace the original infrastructure used to create the application. In this study, we use APIs to access free cloud datastore resources. They also replaced the web server and database. If we use this architecture without a local physical infrastructure, development, maintenance, and human resource costs can be reduced.

A cloud datastore plays a key role in the proposed architecture. Personal user data can be stored in the cloud datastore by replacing the web server. We also integrate the data formats of different cloud service datastores such as Google Datastore, MongoDB, and Hadoop HBase, which are discussed in the next section.

A. Web-based Application

In a web-based application, data is accessed through a web browser. Users do not need to install or configure the application. It can be used on any platform and operating system.

The application is accessed via a browser with a server node called a web server. The browser and server communicate using the HTTP protocol. In this study, we created a web-based application. To compare the developed application with a cloud-based application and determine differences and advantages, the application functions are quite similar.

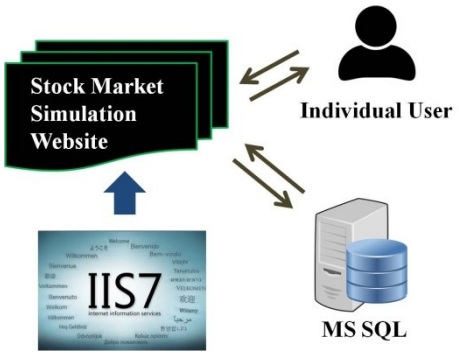

Fig. 4. The architecture of the web-based application

This application was developed using ASP.Net and C#, including CSS, HTML, and other elements. We used the Microsoft SQL Server as the back-end database. ASP.NET is included in the Microsoft .NET framework, which is a set of libraries for web application development. The framework provides functions to process and expand ASP.NET pages. In the present study, we use it to improve availability and ease of connection.

B. Cloud-based Application

A cloud-based application was developed using cloud-based SOA, which is a set of network architectures. This application provides many different network services. Each individual provides network services for other individuals. As mentioned previously, we used cloud-based SOA to construct cloud applications, and we have used MongoDB, Google Cloud Datastore, Hadoop HBase, and their respective APIs for database services. We have also established an agent to handle cross-service integration. The user can connect via an API to communicate to the cloud server. The system transmits messages via the agent to request access to various cloud databases.

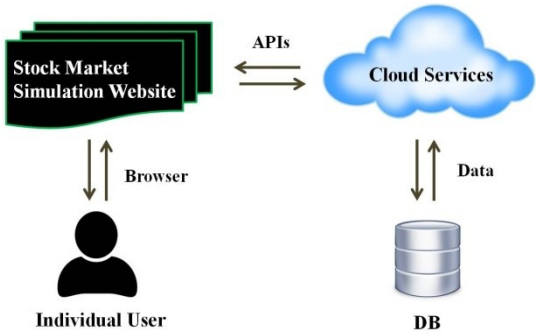

Fig. 5. Architecture of the cloud-based application

1) MongoDB

MongoDB is a document-oriented database system that supports searching and indexing NoSQL databases written in C++. It also supports many programming languages. Because MongoDB is a NoSQL database, some of the basic terms differ from traditional relational database terms. These differences are shown in the following chart:

RDBMS	MongoDB
Database	Database
Table	Collection
Record/Row	Document
Column	Field
Primary Key	_id

MongoDB is the first cloud database we used. We used the MongoDB Server program provided by MongoLab, which allowed us to build a MongoDB for free. Through the MongoDB API, we can communicate with the cloud database and send requests. Note that add, query, modify, and delete functions are allowed. MongoLab is a fully managed cloud database service with high availability. Developers and IT professionals can focus on product development rather than operations by hosting MongoDB on MongoLab's Database-as-a-Service platform [6].

2) Google Cloud Datastore

We used the Google Cloud Datastore API to link to the cloud database. The Google Cloud Datastore is a highly available and durable schemaless NoSQL database that provides robust and scalable storage. We use the Google App Engine as a platform. The code, including API usage, was uploaded to the Google App Engine. The Google App Engine is a platform for building scalable web applications and mobile application back-ends. It provides built-in services and APIs [7]. However, in the integrated system, we will use differences.

3) Hadoop Hbase

HBase is an open-source distributed database that can store structured, semi-structured, and unstructured data. HBase does not employ relational database architecture; thus, it does not support SQL. HBase is specifically designed to operate in a cluster environment. Each node in the cluster provides storage, caching, and operations. This provides HBase with excellent expansion flexibility and fault tolerance. In Hadoop HBase, we use the HBase REST API client libraries and build projects using the Microsoft Azure platform. We use the HBase .NET SDK and other components. Azure is a cloud computing platform provided by Microsoft. Azure provides a cloud operating system, basic storage, and management platform, which are required for online services. In the code, we must first set the platform information to facilitate connection, including URLs, users, and passwords, as follows:

```
string clusterURL = "https://<HBaseClusterName>.azurehdinsight.net";
string hadoopUsername= "<HadoopUsername>";
string hadoopUserPassword = "<HadoopUserPassword>";
```

We can access and communicate with HBase through the Microsoft Azure platform and the HBase REST API.

4) Integrated Cross-Services System

After implementing the connection for the three cloud databases, we attempt to integrate the three cloud spaces to an application. The application runs on Microsoft Azure.

First, we establish a web project in Visual Studio and Azure App Service's Web application. We deploy the website

code project to the Web applications. This allows users to operate the website over the Internet. We can also manage and monitor Azure services through its portal.

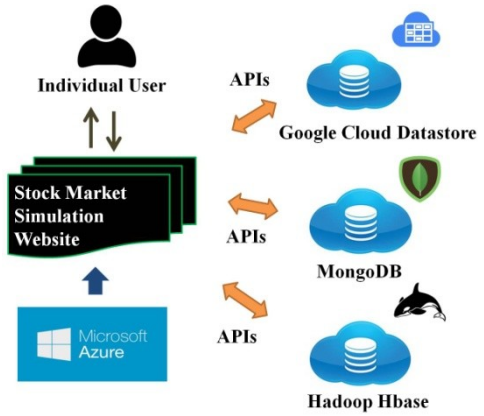

Fig. 6. The architecture of the Integrated Cross-Services System

For data access, we divided the original data among three storage systems, i.e., MongoDB, Google Cloud Datastore, and Hadoop HBase, to test different cloud service connections and demonstrate that the system can be used on these cloud services. In addition, we want to integrate several cloud services. The code to connect to MongoDB and HBase is written directly into the main program using C# and APIs. The Google Cloud Datastore connection is written in Java.

To integrate the three cloud databases into an application, we packed it into a web service to facilitate communication with other applications. We joined the web server project to the ASP.NET service reference inside the program, which enables data exchange and data requests. Web services are software services that communicate with each other via the World Wide Web. They provide services or exchange data with other applications or heterogeneous systems. We also built an agent to distribute and process data storage that can communicate with each cloud database. The storage and use of data is performed using this agent.

C. *Process of Big data*

Big data is a collection of large and complex data sets. Big data is a general term used to describe large amounts of unstructured and semi-structured data. The life cycle of big data can be illustrated with a simple graph. There are four stages in this version of the big data life cycle: Generation, Collect & Store, Analysis & Computation, and Data Collaboration & Sharing [8]. This version of the big data life cycle was proposed by BlazeClan, which is a cloud consulting company in partnership with Amazon Web Services Cloud and Microsoft Azure.

Fig. 7. Big Data life cycle

In this study, data is generated from a stock market. We collected market data in the second stage, which is available online. We created a service called Big Service, which specializes in sending messages and communicating with other services. It can call for requests between services. The Big Service can store data in either a relational database or a cloud datastore. RDBs and cloud datastores have their own storage formats, and Big Service can identify these different services and select the correct format to use.

In the third stage, we identified which data is more suitable for our purpose. The Big Service can assess and execute by condition. In the final stage, we share and merge the market data on our website, which provides a service that simulates a stock market. The service is used in a financial course. The purpose is to provide a simulation that helps users become familiar with the stock market.

In addition, we created a Processing Service to handle orders placed by the user. The process is performed according to a real stock transaction; however, it does not run in real time. The Processing Service handles orders according to daily stock data after classification.

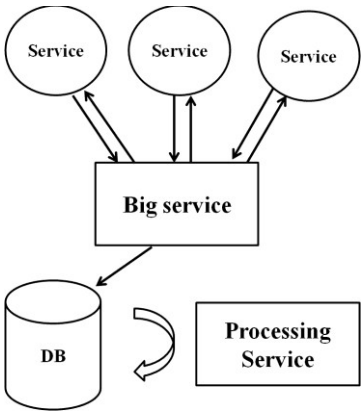

Fig. 8. Big Data process

D. *System analysis*

A use case diagram is a key diagram in the Unified Modeling Language. It is an excellent technique to understand and describe requirements. The application in this research is divided into three main parts, i.e., market data management, transaction record management, and order management. It also extends many functions, such as follow a selected stock and personal information management. The actors we use are an order entry and account information agent, stock transaction agent, market data agent, and user.

The user can use functions, such as searching the market data, viewing transaction records, and order management. The market data agent collects and stores market data in the system database. It can also manage the market data, including add, delete, and modify operations. The stock transaction agent is a primary component of processing transaction data. The order entry and account information agent performs order management and personal user information management.

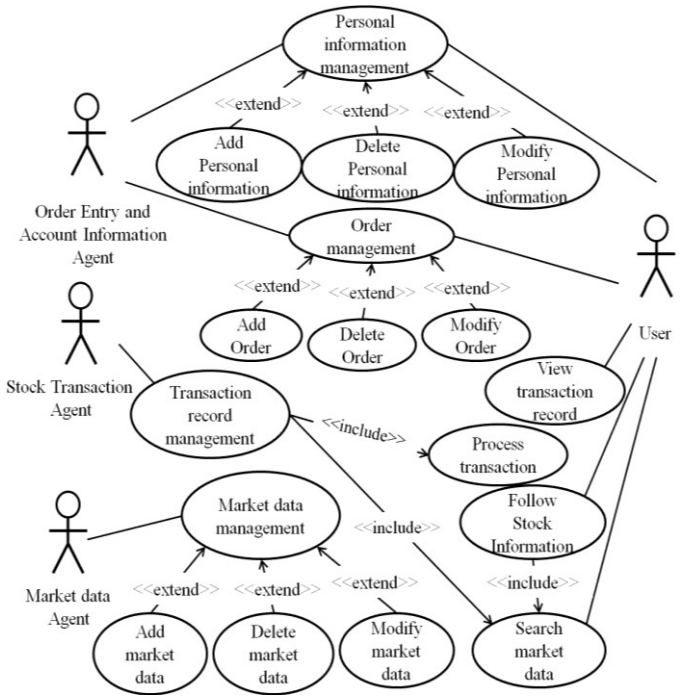

Fig. 9. Use Case Diagram

In use, users need to log in on the beginning page. In our service, users can easily search all the market data which is collected by market data agent. The market data include daily and historical data. They can only type the key word to find out their target data. We also provide an additional function of sorting to help them collating and analyzing data.

Then, the services allow users to place an order that simulate to realistic stock market. When the setting time is up, stock transaction agent will process the orders and judge the results. We set many rules according to the market and try to enhance their similarity.

At last, users could manage their personal and order information through order entry and account information agent. Users can change part of the personal information and look over their records of orders.

To place an order, the agent would record the order first. Then do a series of check before it accept the order. The final stage we set a step for user to reconfirm their order and make sure the accuracy. The order will temporarily place to database while it gets accepted.

IV. SERVICE FEATURES

Figure 10 shows screenshots of our service. The service has the following features:

- Cross service integration and application
- Schemaless NoSQL datastore
- Free cloud resources
- Stock market simulation
- Cloud virtualization and application
- Big data processing (stock and transaction information)
- Cloud services

Fig. 10. A screenshot of Stock Market Simulation. (Interface)

V. EXPERIMENT & EVALUATION

We used Pingdom [9], which is a free website tool, to understand and evaluate the performance of the proposed services. We also compared the performance differences between the web-based and cloud-based applications. We conducted experiments to measure the load times of the service website. We set New York as the test area and disabled the cache to facilitate fair evaluation. The specifications of the server used in our tests are as follows:

A. *Web-based server*

- CPU: Intel® Core(TM) Quad CPU Q9400 @ 2.67GHz.
- RAM: 4GB
- Operating System: Windows Server 2008 R2 Enterprise 64bit
- Located in Taiwan

B. *Cloud-based server*

- Microsoft Azure server
- Located in America

Table I shows the results of the load time test. As can be seen, the cloud-based application is faster than the web-based application. In addition, the difference in terms of response time is significant. We speculate that the reason for this is that cloud service providers specialize in cloud storage. This difference is also influenced by network speed, test location and equipment capabilities between the cloud service provider and our computer facilities.

TABLE I. THE RESULTS OF WEBSITE LOAD TIME TEST

Load Time	Web-based Application (Computer Server)	Cloud-based Application (Microsoft Azure)
1	3.00s	1.40s
2	3.08s	1.55s
3	2.72s	1.43s
4	2.48s	1.31s
5	2.72s	1.20s
6	2.81s	1.59s
7	2.93s	1.41s
8	2.82s	1.35s
9	3.01s	1.48s
10	2.77s	1.22s

In addition, we used Load Impact to perform load testing. Load Impact is a leading online load and performance testing service. It uses Amazon cloud rooms all over the world to perform load testing. It can simulate many users in different countries effectively [10]. We set the area to Ashburn, USA (Amazon) and Tokyo, Japan. By comparing the load test results, we can understand the scalability of different types of applications. We conducted stress tests on the web-based application (hosted in our computer facility) and the cloud-based application.

The Load Impact tool was configured with 100 users accessing the website simultaneously over a period of five minutes. The number of users was increased gradually. The results show the VUs Active and the VU Load Time of the server, respectively.

We know the test position is closer to web-based application in JP but closer to cloud-based application in USA. The load times of the web-based application in USA and JP were approximately 3.5 s and 0.8 s, cloud-based application in USA and JP were approximately 0.5 s and 1.3 s. This indicates that both applications are stable and have good pressure resistance. The only difference is the disparity of the load time. Test position can affect the results. But on average, it is faster and better to use a cloud service as a website server. We cannot know the details and specifications of the Microsoft server; however, we can confirm that cloud service providers offer resources that are significantly better than building our own system.

VI. Conclusion & Future Work

Many free services on the Internet provide various APIs for developers to create applications. This study used a number of APIs to complete the structure. They also contain a number of functions and combine real-world stock markets. Not only in the integration platform but also completed in the stock market simulation to prove that this study can be practical applied. There are many different future development directions for the proposed architecture, which are outlined in the following.

A. Different Access Method

In this study, we used our personal cloud as the cloud space for the proposed architecture. Applications communicate with a cloud datastore in our personal cloud through APIs. Another way is to obtain user consent to access the user's account through the application. By using the OAuth authorization process in third-party, applications can access user resources on a site without providing a username and password to third-party applications.

This will change the access position and method of the present study. If this method is used, data will be stored in each user's personal cloud storage rather than a centralized repository. It would less likely need to remove part of the file caused by the problem due to too much data occupancy. Its disadvantage is that data cannot be managed centrally if a problem occurs or data is tampered with. This issue also includes a number of information security issues, and addressing such issues is a significant challenge.

B. Data Storage Method

In this study, all data was divided among three cloud databases, and each database stored different data. Another method is to store all data in each cloud database for backup and redundancy purposes. However, this would increase resource requirements.

We must also decide which cloud service to use. In addition, synchronization issues also appear, and a heartbeat may be a solution to this issue. Heartbeats, which are used in multi-node systems, are signals that are generated at regular intervals to determine whether something is working correctly. We must set one machine as the primary node and the other as the secondary node. If the primary node fails or shuts down, the secondary node can perform the functions of the primary node [11]. However, heartbeats have many restrictions relative to operating systems and programs. In future, we may investigate solutions to address these problems.

We have successfully built a service that can support various cloud datastores. This study used integrated datastores. In addition, the APIs provided by cloud services are limited. For example, many APIs or services are updated or revised, which means that people using free cloud resource would need to revise code more frequently. We will investigate these limits in future. Furthermore, although this research has applied in practice, it is not sufficiently mature to be operated stably over long periods. We intend to address these issues in the near future.

References

[1] S.E. Chang, K.-M. Chiu , and Y.-C. Chiao, "Cloud Migration: Planning Guidelines and Execution Framework," Proc. 2015 Seventh Int. Conf. Ubiquitous and Future Networks, pp. 814-819, IEEE, 2015.

[2] G. Breiter and V.K. Naik, "A Framework for Controlling and Managing Hybrid Cloud Service Integration," 2013 IEEE Int. Conf. Cloud Eng., pp. 217–224, IEEE, 2013.

[3] V. Beal, "Cloud Services," Webopedia, Retrieved 15 Nov. 2015. Available at: http://www.webopedia.com/TERM/C/cloud_services.html.

[4] C. Xue and Z. Zeng, "The Construction of SOA-Based Enterprise Information Management System," Proc. 2010 International Conference on Computer Application and System Modeling, vol. 6, pp. 522–525, IEEE, 2010.

[5] F. Lo, "Bigdata Technology—What is Hadoop? What is NoSQL? What is MapReduce?" Retrieved 20 Nov. 2015. Available at https://datajobs.com/what-is-hadoop-and-nosql

[6] MongoDB Inc., "MongoDB for GIANT Ideas | MongoDB." Retrieved 5 Jan. 2016. Available at https://www.mongodb.com

[7] Google, "Datastore–NoSQL Schemaless Database, Google Cloud Platform." Retrieved 13 Oct. 2015. Available at https://cloud.google.com/datastore/

[8] BlazeClan Inc., "Build Your Big Data Building Blocks with AWS Cloud." Retrieved 5 Jan. 2016. Available at http://www.blazeclan.com/build-your-big-data-building-blocks-with-aws-cloud/

[9] Pingdom, "Pingdom Website Speed Test." Retrieved 22 Dec. 2015. Available at http://tools.pingdom.com/fpt/

[10] Load Impact, "On Demand Load Testing for Developers & Testers." Retrieved 21 Dec. 2015.. Available: at https://loadimpact.com

[11] M. Rouse, "Heartbeat," TechTarget. Retrieved 22 Dec. 2015. Available at http://searchenterpriselinux.techtarget.com/definition/Heartbeat